Improved Error Bounds Based on Worst Likely Assignments

Eric Bax
Email: baxhome@yahoo.com

arXiv:1504.00052v1 [stat.ML] 31 Mar 2015

***Abstract*—Error bounds based on worst likely assignments use permutation tests to validate classifiers. Worst likely assignments can produce effective bounds even for data sets with 100 or fewer training examples. This paper introduces a statistic for use in the permutation tests of worst likely assignments that improves error bounds, especially for accurate classifiers, which are typically the classifiers of interest.**

I. Introduction

Permutation tests are used in statistics for hypothesis testing, especially in bioinformatics [1], [2], [3], [4], [5]. Permutation tests apply to a wide range of problems because they do not rely on assumptions about the form of the distributions that generate samples. Because they do not rely on asymptotic results, they are ideal for small-sample problems. As a result, permutation tests are often used in exact tests [6], [7].

Error bounds based on worst likely assignments [8], [9] incorporate permutation tests as part of a technique to produce error bounds for classifiers. Worst likely assignments can produce effective error bounds even for small data sets. In this paper, we show effective error bounds even for data sets with 100 or fewer training examples.

Worst likely assignment error bounds apply to the transductive setting [10], where there is a set of training examples with known inputs and labels, and a set of working examples with known inputs and unknown labels. The goals are training—developing a classifier that performs well on the working examples—and validation—producing a bound on the classifier's error rate over the working examples.

The worst likely assignment technique tests each potential assignment of class labels to the working examples, evaluating whether the assigned labels cause the working examples to "blend in" among the training examples. If so, then the assignment is declared a likely assignment, making the classifier's error rate given the assignment a candidate for the error bound. If not, then the assignment is dismissed as being unlikely.

This paper presents a new type of statistic for use in the permutation test to determine whether an assignment is likely. The statistic is a *scoring function* that evaluates whether assigned labels cause working examples to have rates of disagreement with neighboring examples' labels that are similar to those rates for training examples. The scoring function significantly improves error bounds for accurate classifiers.

This paper is organized as follows. Section 2 defines terms and notation. Section 3 reviews error bounds based on worst likely assignments. Section 4 presents the new scoring function. Section 5 demonstrates how well the scoring function performs on some data sets. Section 6 closes with a discussion of challenges for future work.

II. Concepts and Notation

This paper concerns validation of classifiers learned from examples. Each example $Z = (X, Y)$ includes an input X and a class label $Y \in \{0,1\}$. Define *complete sequence* C to be a random variable:

$$C = Z_1, \ldots, Z_{t+w}$$

with examples $Z_1 = (X_1, Y_1)$, ..., $Z_{t+w} = (X_{t+w}, Y_{t+w})$ drawn i.i.d. from an unknown joint distribution D of inputs and labels. We observe

$$(X_1, Y_1), ..., (X_t, Y_t), X_{t+1}, ..., X_{t+w}$$

that is, inputs and outputs of t training examples and just the inputs of w working examples. A classifier g, which is a mapping from the input space of X to $\{0,1\}$, is developed using the observed data. Then classifier g is used to predict the working example outputs Y_{t+1}, ..., Y_{t+w} associated with inputs X_{t+1}, ..., X_{t+w}.

For any sequence of examples

$$c = (x_1, y_1), \ldots, (x_{t+w}, y_{t+w})$$

from the joint space of inputs and labels, define the *error* to be

$$E_c = \frac{1}{w} \sum_{i=t+1}^{t+w} I(g(x_i) \neq y_i),$$

where I is the indicator function—one if the argument is true and zero otherwise. The goal is to produce a PAC (probably approximately correct) bound on E_C, the error on complete sequence C.

III. Worst Likely Assignment Error Bound

Use σc to denote the sequence c permuted by permutation σ of $1, \ldots, t+w$. Use σ^{-1} to denote the inverse of permutation σ, such that $\sigma\sigma^{-1}c = c$ for all c. Let $h()$ be a real-valued *scoring function* on sequences of $t+w$ examples. For example, $h(c)$ could be the error rate over the last w inputs for a classifier trained on the first t examples in c. Let Q be a set or multi-set of permutations of $\{1, \ldots, t+w\}$. Define the *ranking function* $r(x, S)$ to be the rank of value x among the entries

in set or multiset S, with random tie-breaking to determine the rankings among equal values.

Theorem 1. *If C is drawn according to D^{t+w} and σ^* is drawn uniformly at random from Q, then*

$$\forall k \in \{1, \ldots, |Q|\} :$$

$$Pr[r(h(C), \{h(\sigma\sigma^{*-1}C)|\sigma \in Q\}) = k] = \frac{1}{|Q|}$$

where the probability is over random draws of C and σ^*. In other words, if C is mapped to a random permutation in Q, then C is equally likely to have each rank among permutations of C relative to the mapped permutation.

Proof: Since the elements of C are i.i.d., for each sequence c, each permutation of c is equally likely to be C. So the distribution of $\sigma^{*-1}C$ is the same as the distribution of C, and we can replace $\sigma^{*-1}C$ by C in the theorem to get the logically equivalent statement:

$$\forall k \in \{1, \ldots, |Q|\} :$$

$$Pr[r(h(\sigma^*C), \{h(\sigma C)|\sigma \in Q\}) = k] = \frac{1}{|Q|},$$

where the probability is over random draws of C and σ^*. To prove this equation, it is sufficient to show that

$$\forall c : \forall k \in \{1, \ldots, |Q|\} :$$

$$Pr[r(h(\sigma^*c), \{h(\sigma c)|\sigma \in Q\}) = k] = \frac{1}{|Q|},$$

where the probability is over random draws of σ^*. Since σ^* is selected uniformly at random from Q, $h(\sigma^*c)$ is equally likely to be each entry in

$$\{h(\sigma c)|\sigma \in Q\}.$$

So $h(\sigma^*c)$ is equally likely to have each rank in $1, \ldots, |Q|$. ∎

Let $a = (\hat{y}_{t+1}, \ldots, \hat{y}_{t+w})$ denote an assignment to the (unknown) outputs of the working examples in sequence C, and let $C(a)$ denote the sequence with the values in a assigned to the working example outputs. Let $a^* = (y_{t+1}, \ldots, y_{t+w})$ be the actual outputs, so that $C(a^*) = C$. For a given *bound failure probability* δ, define a *likely set* of assignments:

$$L = \{a \in \{0,1\}^w|$$

$$r(h(C(a)), \{h(\sigma\sigma^{*-1}C(a))|\sigma \in Q\}) \leq \lceil (1-\delta)|Q| \rceil\}.$$

In other words, the likely set contains the assignments a that rank in the bottom $1 - \delta$ in the test: assign a to the unknown outputs of the working set of C, permute $C(a)$ by σ^{*-1}, take all permutations σ in Q of $\sigma^{*-1}C(a)$, compute their scores $h(\sigma\sigma^{*-1}C(a))$, then check the rank of the score of $h(C(a))$ among the scores.

Theorem 2. *If C is drawn according to D^{t+w} and σ^* is drawn uniformly at random from Q, then*

$$Pr[E_C \leq \max_{a \in L} E_{C(a)}] \geq 1 - \delta,$$

where the probability is over random draws of C and σ^*.

Proof: Since $C = C(a^*)$,

$$E_C = E_{C(a^*)}.$$

So

$$a^* \in L \implies E_C \leq \max_{a \in L} E_{C(a)}.$$

Examine the definition of L. When $a = a^*$, $C(a) = C$. So

$$r(h(C), \{h(\sigma\sigma^{*-1}C|\sigma \in Q\}) \leq \lceil (1-\delta)|Q| \rceil \implies a^* \in L.$$

According to Theorem 1,

$$r(h(C), \{h(\sigma\sigma^{*-1}C|\sigma \in Q\})$$

is equally likely to have each value in $1, \ldots, |Q|$. So the probability that the value is in $1, \ldots, \lceil (1-\delta)|Q| \rceil$ is at least $1 - \delta$. ∎

Error bounds based on Theorem 2 are called worst likely assignment error bounds because the bound is the highest error that is consistent with a likely assignment. Theorem 2 is general. A specific error bound requires a scoring function $h()$ and a permutation set Q. Effectiveness of the error bound and ease of computation influence the selection and design of scoring functions and permutation sets. Bax and Callejas [8] outline several scoring functions, including the error from developing a classifier based on the first t examples and applying it to the last w examples from the argument sequence of $t + w$ examples. They also also introduce two types of permutation sets: the set of all permutations and random subsets of permutations. This paper introduces a new scoring function that improves error bounds.

IV. A Near Neighbor Scoring Function

Consider a scoring function that counts differences between labels of examples in the last w examples and their nearest neighbors in the first t examples of the sequence c being scored. Suppose that nearby neighbors among the training examples tend to have the same labels. Then the scores will typically be low when scoring permutations of c, because many of the examples in the last w of the permuted sequence will be training examples, and in many cases their nearest neighbors in the first t examples of the permuted sequence will be nearby neighbors among the training examples. These low scores constrain the set of likely assignments to those with high rates of agreement between labels assigned to the working examples and the labels of their nearest neighbors among the training examples. As a result, the scoring function typically produces strong error bounds.

This section defines a class of scoring functions that count disagreements between examples in the last w examples and their nearby neighbors in the first t examples. The disagreements can be summed over multiple near neighbors to make the scores more robust. Also, the contributions to the sum from different neighbors can be weighted to emphasize nearer neighbors more.

Let

$$n_{ijS}(c)$$

be the label of the ith nearest neighbor to example j in sequence $c = \{(x_1, y_1), \ldots, (x_{t+w}, y_{t+w})\}$ among the examples indexed by set S, with random tie-breaking. Define a class of scoring functions

$$f_{\alpha kS}(c) = \sum_{i=1}^{k} \sum_{j \in \{t+1,\ldots,t+w\}} \alpha^{i-1} I(n_{ijS} \neq y_j) \qquad (1)$$

where the indicator function $I()$ is one if the argument is true and zero otherwise.

These scoring functions count disagreements between the last w examples in c and their nearest neighbors among a subset of examples in c. The parameter α specifies how much to weigh disagreements with nearby neighbors relative disagreements with more distant neighbors. The parameter k expresses how many nearby neighbors to consider. Set S constrains the examples for which disagreements can be counted.

Setting $S = \{1, \ldots, t\}$ specifies that when the scoring function is applied to the sequence consisting of training examples followed by working examples with assigned labels, the score ignores disagreements between pairs of working examples, which both have assigned labels. So the scoring function does not favor assignments with incorrect, but agreeing, labels assigned to neighboring working examples. Instead, the score is based solely on disagreements between pairs of examples that have one working example with an assigned label and one training example with an actual label drawn by sampling. So it favors assignments that label working examples similarly to their neighboring training examples.

V. Tests

This section presents results from applying the near neighbor scoring function to produce error bounds for several data sets. For each data set, the results include a figure showing how adjusting the influence of more distant neighbors affects error bounds and a table showing more detailed statistics for the error bounds for a few parameter settings from the figure. For each data set, there are comparisons over a range of bound certainty values δ.

In these tests, NNSF refers to the near neighbor scoring function $f_{\alpha kS}$ defined in Equation 1. The parameter settings are $\alpha \in \{0.0, \ldots, 0.9\}$ for the figures, $\alpha = 0.5$ for the tables, and $k = t$ and $S = \{1, \ldots, t\}$ for both figures and tables. So in these tests, NNSF weighs each disagreement with a neighbor by α as much as disagreement with the next nearest neighbor, and, for each example in the last w examples, disagreements with examples in the first t examples contribute to the score. In the figures, as α increases from zero to one, more neighbors play significant roles in determining whether to accept or reject an assignment.

ESF (for error scoring function) refers to a baseline scoring function that uses the error from using the first t examples as a 1-nearest neighbor classifier on the last w examples. This scoring function, which was introduced by Bax and Callejas [8], is equivalent to $f_{\alpha kS}$ with $k = 1$ and $S = \{1, \ldots, t\}$. (We treat 0.0^0 as one in Equation 1, so ESF is also equivalent to NNSF with $\alpha = 0.0$.)

For each figure, each line shows results for a different value of the bound certainty parameter δ. The value of α varies along each line. The plotted amounts are differences between error bound and actual error rate. Each plotted value is an average over 1000 trials.

For each table, each row holds results for a different value of the bound certainty parameter δ. The second column of each table shows errors from using training data as a 1-nearest neighbor classifier on working data. The subsequent columns show differences between the bounds on error and actual error for bounds using scoring functions NNSF and ESF.

Each cell shows a mean and standard deviation over 1000 trials. The cells in the "Error" column show mean and standard deviation of errors. The cells in subsequent columns show mean and standard deviation of difference between bound and error. For example, suppose the error has mean 0.3 and standard deviation 0.4, and a bounding method has mean 0.1 and standard deviation 0.0. This indicates that the error averages 0.3 over the 1000 trials, and the error varies quite a bit, but the bound is always exactly 0.1 greater than the actual error.

Note that the standard deviations displayed in cells are standard deviations of the values over 1000 trials. They are not standard deviations of the estimates of the means of values over 1000 trials, that is, their large sizes do not indicate uncertainty about the accuracy of the means. Since there are 1000 trials, those standard deviations are about 1/33 of the ones shown, indicating that most differences in means of bounds produced by the different scoring functions are statistically significant for most of the tests.

Each figure line and table row is based on the same 1000 trials, but different lines and rows are based on different sets of trials. For each trial, a size *t+w* subset of examples is selected at random from a data set. A size-*t* subset is selected at random to form the training set, and the remaining *w* examples form the working set. The error is computed, and error bounds are computed using the scoring functions NNSF and ESF. The error is subtracted from each bound, and the differences are accumulated into the statistics shown in the figures and tables.

A. Iris Data

Figure 1(a) and Table I show results for a data set involving iris classification. The data set is from the repository of data

TABLE I
IRIS DATA – COMPARING SCORING FUNCTIONS

δ	Error	NNSF - Error	ESF - Error
0.025	0.000±0.000	0.249±0.016	0.408±0.121
0.050	0.000±0.000	0.235±0.060	0.296±0.104
0.075	0.000±0.000	0.154±0.122	0.247±0.084
0.100	0.000±0.000	0.017±0.063	0.213±0.096
0.125	0.000±0.000	0.000±0.008	0.182±0.113
0.150	0.000±0.000	0.000±0.000	0.142±0.124

TABLE II
LINEAR CLASS BOUNDARIES – COMPARING SCORING FUNCTIONS

δ	Error	NNSF - Error	ESF - Error
0.100	0.067±0.078	0.114±0.097	0.194±0.100
0.200	0.073±0.083	0.051±0.109	0.118±0.101
0.300	0.068±0.079	0.010±0.106	0.066±0.097

sets for machine learning maintained by the University of California at Irvine, which is available online. The data set contains examples for three types of iris; we use only the examples for the first two types in order to produce binary classification problems. This leaves 100 examples, with 50 from each class. Each example has four input dimensions. We use t = 40 training examples and w = 4 working examples for each trial. The classification method is 1-nearest neighbor. The iris data are easy to classify, as indicated by the fact that the errors are always zero.

For both scoring functions, the bounds use as Q a set of permutations that makes each size 4 subset of the 44 examples the last 4 examples in the sequence exactly once. (This is equivalent, except for random tie-breaking, to using all 44! permutations of the sequence of examples as Q.) Bax and Callejas call this a *complete filter* [8].

Figure 1(a) shows that, for most values of δ, the bound improves substantially from $\alpha = 0.0$ to $\alpha = 0.1$, which goes from considering only agreement with the nearest neighbor among training examples to evaluate the label of each working example to giving other nearby neighbors a role as well. Then the bounds stay about the same as α increases.

Table I shows that NNSF with $\alpha = 0.5$ has a statistically significant advantage over ESF over a full range of δ values. For $\delta = 0.1$ and greater, NNSF produces bounds that are at least an order of magnitude tighter than those produced by ESF. For $\delta = 0.15$, NNSF returned a bound that is equal to the actual error rate of zero for all 1000 problems.

NNSF performs so well because the classes in the iris data are well-separated. The nearest several neighbors to an example are usually all from the same class as the example. As a result, NNSF easily rejects as unlikely those assignments that mislabel working examples, because their several closest neighbors among the training examples all disagree with the incorrect label.

B. Data with a Linear Class Boundary

Figure 1(b) and Table II show results for randomly generated data. The data consist of 1100 examples drawn uniformly at random from a three-dimensional input cube with length one on each side. The class label is zero if the input is from the left half of the cube and one if the input is from the right half of the cube. For these tests, there are t = 100 training examples and w = 10 working examples, using 1-nearest neighbor classification.

For both scoring functions, the bounds use as Q a set of 1000 permutations drawn uniformly at random without replacement from a set of permutations that makes each size 10 subset of the 110 examples the last 10 examples exactly once. Bax and Callejas call this a *sample filter* [8]. Sampling is used to reduce computation.

Figure 1(b) shows that, as for the iris data, the largest improvement in the bounds for data with a linear class boundary comes from changing α from 0.0 to 0.1, to give multiple neighbors a role in evaluating assignments. From there, increasing α produces stronger bounds until $\alpha = 0.4$ or $\alpha = 0.5$. Then, increasing α more produces weaker bounds, as more distant neighbors that are less likely to be from the same class as the working example being evaluated play a stronger role in the scoring function.

Table II shows that NNSF with $\alpha = 0.5$ produces statistically significantly better bounds than ESF for this data. The ratio of the difference between bound and error for NNSF to that of ESF increases as bound certainty parameter δ increases. NNSF performs well for this data because, as for the iris data, the nearest neighbors to each working example among the training examples are usually from the same class as the working example. As a result, NNSF rejects as unlikely most assignments that mislabel working examples.

TABLE III
NONLINEAR CLASS BOUNDARIES – COMPARING SCORING FUNCTIONS

δ	Error	NNSF - Error	ESF - Error
0.100	0.186±0.129	0.242±0.141	0.245±0.145
0.200	0.180±0.125	0.155±0.151	0.160±0.143
0.300	0.177±0.122	0.105±0.157	0.116±0.138

C. Data with a Nonlinear Class Boundary

Figure 1(c) and Table III show results for randomly generated data with a nonlinear class boundary. The data have the same characteristics as in the previous test, except that each class label is determined by the XOR of whether the input is in the left half of the cube, the bottom half of the cube, and the front half of the cube. In other words, the cube is cut into eight sub-cubes, and each sub-cube has a different class than the three sub-cubes with which it shares a side. This class scheme introduces more error than for the data with a linear class boundary. Similar to the data with linear class boundaries, 1-nearest neighbor classification is used, and the bound method uses a random sample of 1000 permutations as Q.

Figure 1(c) shows results similar to those for data with a linear class boundary, except that increasing α beyond 0.2 produces weaker bounds. This occurs for lower α with this data set, because working examples in this data set are more

TABLE IV
PIMA INDIAN DIABETES DATA – COMPARING SCORING FUNCTIONS

δ	Error	NNSF - Error	ESF - Error
0.100	0.322±0.136	0.252±0.139	0.225±0.147
0.200	0.309±0.135	0.197±0.140	0.162±0.146
0.300	0.314±0.139	0.154±0.150	0.108±0.151

likely to have some near neighbors from a different class than working examples in the data set with a linear class boundary.

For the data with nonlinear class boundaries, Table III shows that NNSF with $\alpha = 0.5$ and ESF perform similarly well. For $\delta = 0.1$ and 0.2, the differences in bounds produced by NNSF and by ESF are not statistically significant. For $\delta = 0.3$, the difference is statistically significant, but small. For this data, working examples are likely to have near neighbors among the training examples that are from a different class. As a result, NNSF does not have a strong advantage over ESF.

D. Pima Indian Diabetes Data

Figure 1(d) and Table IV show results for data related to diabetes among Pima Indians. The data set is available from the online repository maintained by the University of California at Irvine. The data set has 768 examples: 500 from one class and 268 from another. Each example has eight input dimensions. Since the input dimensions have different scales, we normalize the data, translating and scaling each input dimension to give it mean zero and standard deviation one. We use t = 200 training examples and w = 12 working examples for each trial, with Q a random sample of 100 permutations. The tests use 1-nearest neighbor classification.

The results in Figure 1(d) are similar to those for random data with a nonlinear class boundary. Like that data, working examples in the Pima Indian data are likely to have near neighbors from a different class. Still, as for the nonlinear class boundary data, there is some improvement from using small positive values of α rather than $\alpha = 0.0$. These data are difficult to classify, as shown by the high error rates in Table IV. For this data, ESF outperforms NNSF with $\alpha = 0.5$. However, smaller values of α make NNSF an improvement over ESF even for this data set, as shown in Figure 1(d).

VI. DISCUSSION

This paper developed a new scoring function for worst likely assignment error bounds. For each of our example data sets, the scoring function improved error bounds for some setting of the parameter α, and it improved the error bounds for a blind choice of α for the data sets that produced accurate classifiers. For the accurate classifiers, the method produced effective error bounds even with 100 or fewer training examples.

One challenge for the future is to develop faster methods to compute worst likely assignment error bounds. We can speed up the permutation test for each assignment by sampling permutations, as we did in the tests. It may be possible to use fewer permutations by a cleverer method of sampling (we used uniform sampling) or by fitting the sampled results to a distribution, as in [3].

The greater challenge is to avoid explicitly running a permutation test for each assignment, since the number of assignments is exponential in the number of working examples. For 1-nearest neighbor classifiers, there is a dynamic programming method that computes a worst likely assignment error bound without explicit computation for each assignment [8]. The method produces an error bound in time polynomial in the number of in-sample examples. It would be very useful to extend that method to other types of classifiers, and selecting the right scoring functions may be the key. (An alternative method to achieve polynomial-time computation is to partition the working set, produce an error bound for each partition, and use a union bound over the partitions [9]. But this approach produces looser bounds than validation over the whole working set at once.)

Another challenge is to improve worst likely assignment error bounds for network classifiers [13], [14], [15], [16]. Since many networks grow by accretion, for example friends inviting friends to join social networks, the nodes are not necessarily drawn i.i.d. This makes developing permutation tests for the nodes challenging. However, some subsets of nodes may be drawn i.i.d. [17], allowing permutation tests within those subsets. There is some work using permutation tests for hypothesis testing in social networks [18], [19], [20], [21] and some work on applying worst likely assignments to produce error bounds for them [9]. The challenge for the future is to develop scoring functions specifically designed for network classifiers in order to improve those bounds.

Finally, it would be interesting to explore the concept of best error bounds. Just as there are established criteria for best estimators in statistics (see e.g. [22] pp. 200–202), there should be reasonable criteria for best error bounds. These criteria may drive the discovery and development of new scoring functions.

REFERENCES

[1] P. Golland, F. Liang, S. Mukherjee, and D. Panchenko. Permutation tests for classification. In *P. Auer and R. Meur (Eds.): COLT 2005, LNAI 3559*, pages 501–515, 2005.

[2] C. D. Corcoran, P. Senchaudhuri, C. R. Mehta, and N. R. Patel. Exact inference for categorical data. *Encyclopedia of Biostatistics*, 2005.

[3] T. A. Knijnenburg, L. F. A. Wessels, M. J. T. Reinders, and I. Shmulevich. Fewer permutations, more accurate p-values. *Bioinformatics*, 25:161–168, 2009.

[4] B. Efron and R. Tibshirani. On testing the significance of sets of genes. *Annals of Applied Statistics*, 1:107–129, 2009.

[5] A. Subramanian, P. Tamayo, V. K. Mootha, S. Mukherjee, B. L. Ebert, M. A. Gillette, A. Paulovich, S. L. Pomeroy, T. R. Golub, E. S. Lander, and J. P. Mesirova. Gene set enrichment analysis: a knowledge-based approach for interpreting genome-wide expression profiles. *Proceedings of the National Academy of Science, USA*, 102:1554515550, 2005.

[6] S. Weerahandi. *Exact Statistical Method for Data Analysis*. Springer-Verlag, 1995.

[7] R. A. Fisher. *Statistical methods for Research Workers*. Oliver and Boyd, 1954.

[8] E. Bax and A. Callejas. An error bound based on a worst likely assignment. *Journal of Machine Learning Research*, 9:581–613, 2008.

[9] J. Li, A. Sonmez, Z. Cataltepe, and E. Bax. Validation of network classifiers. *Structural, Syntactic, and Statistical Pattern Recognition Lecture Notes in Computer Science*, 7626:448–457, 2012.

[10] V. Vapnik. *Statistical Learning Theory*. John Wiley & Sons, 1998.

[11] E. Bax. Nearly uniform validation improves compression-based error bounds. *Journal of Machine Learning Research*, 9:1741–1755, 2008.

[12] E. Bax. Validation of average error rate over classifiers. *Pattern Recognition Letters*, pages 127–132, 1998.

[13] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Gallagher, and Tina Eliassi-Rad. Collective classification in network data. *AI Magazine*, 29(3), 2008.

[14] Lise Getoor and Ben Taskar. *Introduction to Statistical Relational Learning*. The MIT Press, 2007.

[15] E. D. Kolaczyk. *Statistical Analysis of Network Data*. Springer, 2010.

[16] Ben London, Bert Huang, and Lise Getoor. Improved generalization bounds for large-scale structured prediction. In *NIPS Workshop on Algorithmic and Statistical Approaches for Large Social Networks*, 2012.

[17] E. Bax, J. Li, A. Sonmez, and Z. Cataltepe. Validating collective classification using cohorts. *NIPS Workshop on Frontiers of Network Analysis: Methods, Models, and Applications*, 2013.

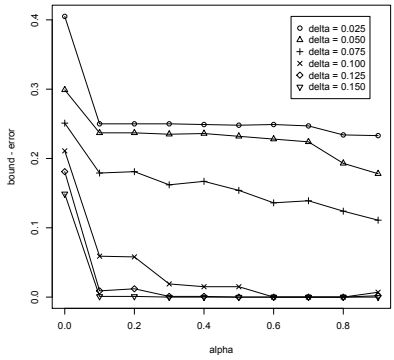
[18] R. Belo and P. Ferriera. Is social influence always positive? evidence from a very large mobile network. *NBER Summer Institute 2013, Economics of Information Technology and Digitization Workshop*, 2013.

[19] R. Belo and P. Ferriera. Using randomization methods to identify social influence in mobile networks. *SocialCom 2012: The Fourth IEEE International Conference on Social Computing*, 2012.

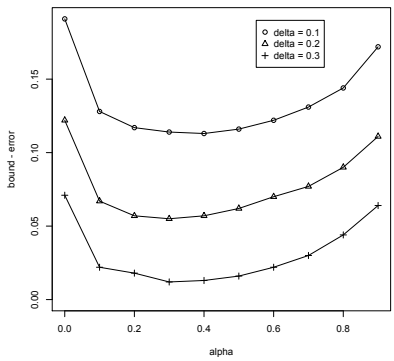
[20] T. La Fond and J. Neville. Randomization tests for distinguishing social influence and homophily effects. *WWW 2010: Proceedings of the 19th International Conference on the World Wide Web*, pages 601–610, 2010.

[21] A. Anagnostopoulos, R. Kumar, and M. Mahdian. Influence and correlation in social networks. *KDD '08: Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 7–15, 2008.

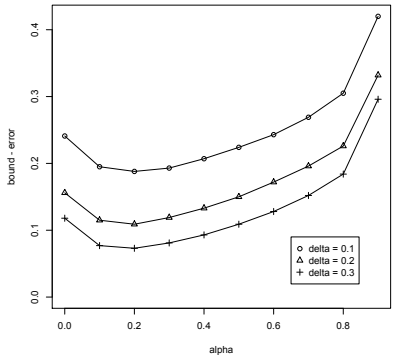
[22] P. G. Hoel. *Introduction to Mathematical Statistics*. Wiley, 1954.



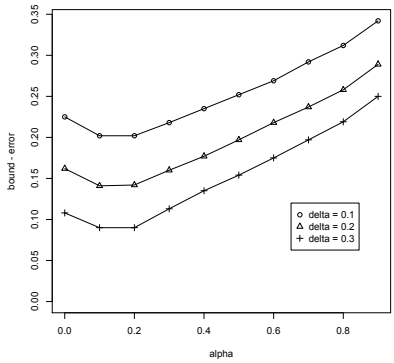
(a) Iris Data



(b) Linear Data



(c) Nonlinear Data



(d) Pima Indian Data

Fig. 1. Bound effectiveness as a function of α.